Fonction mystère

```
def mystere (t,n):
  i=0
  j= n-1
  while(i < j):
    if (t[i] == 0):
      i= i+1
    else :
      echanger(t,i,j)
      j= j-1
```

Que fait-elle en supposant que le tableau T ne contient que des 0 et des 1

Tri rapide

Fonction mystère

```
def mystere (t,n):
  i=0
  j= n-1
  while(i < j):
    if (t[i] == 0):
      i= i+1
    else :
      echanger(t,i,j)
      j= j-1
```

Que fait-elle en supposant que le tableau T ne contient que des 0 et des 1

Fonction mystère

```
def mystere (t,n):
    i=0
    j= n-1
    while(i < j):
        if (t[i] == 0):
            i= i+1
        else :
            echanger(t,i,j)
            j= j-1
```

Fonction mystère

```
def mystere (t,n):
    i=0
    j= n-1
    while(i < j):
        if (t[i] == 0):
            i= i+1
        else :
            echanger(t,i,j)
            j= j-1
```

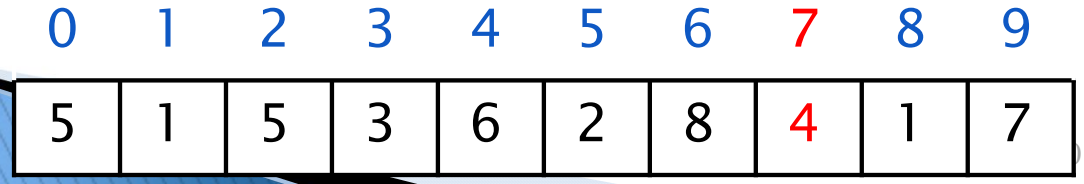
Elle place les 0 au début du tableau et les 1 à la fin

Fonction mystère

Idée :

- Utiliser la fonction `mystere` pour séparer les éléments d'un tableau selon qu'ils sont inférieurs ou supérieur et égaux à une valeur x.

1. Choisir x parmi les éléments de t : donc il existe i ∈[0,n-1] tel que t[i]==x.

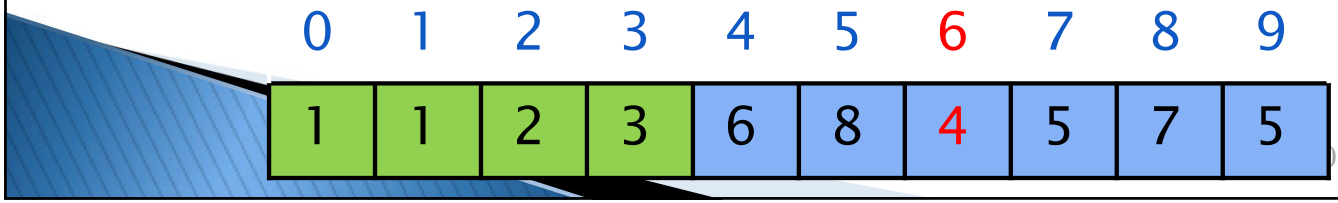


Fonction mystère

Idée :

- Utiliser la fonction `mystere` pour séparer les éléments d'un tableau selon qu'ils sont inférieurs ou supérieur et égaux à une valeur x.

1. Choisir x parmi les éléments de t : donc il existe i ∈[0,n-1] tel que t[i]==x.
2. Séparer les éléments de t en 2 zones par rapport à x.

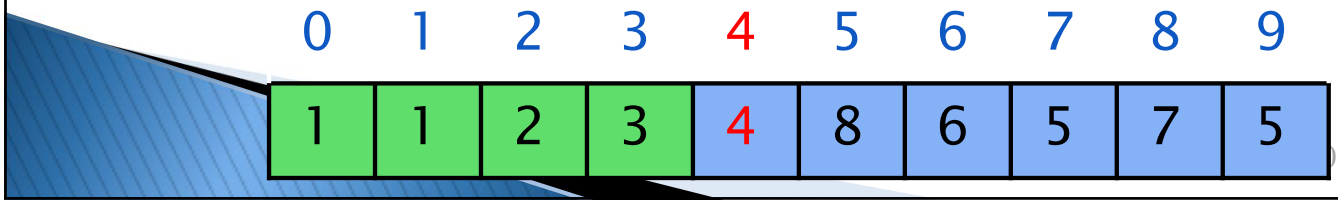


Fonction mystère

Idée :

- Utiliser la fonction `mystere` pour séparer les éléments d'un tableau selon qu'ils sont inférieurs ou supérieur à une valeur x.

1. Choisir x parmi les éléments de t : donc il existe i ∈[0,n-1] tel que t[i]==x.
2. Séparer les éléments de t en 2 zones par rapport à x.
3. Placer x entre les 2 zones.



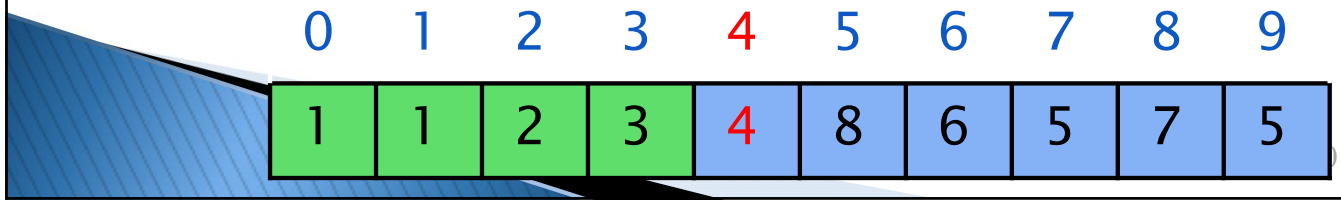
Fonction mystère

Idée :

- Utiliser la fonction `mystere` pour séparer les éléments d'un tableau selon qu'ils sont inférieurs ou supérieur à une valeur x.

1. Choisir x parmi les éléments de t : donc il existe i $\in$[0,n-1] tel que t[i]==x.
2. Séparer les éléments de t en 2 zones par rapport à x.
3. Placer x entre les 2 zones.

Que peut-on dire de x ?



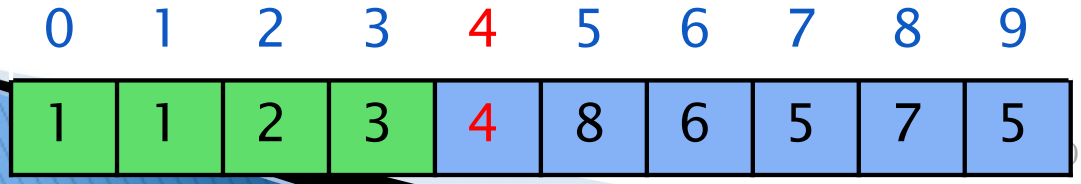
Fonction mystère

Idée :

- Utiliser la fonction `mystere` pour séparer les éléments d'un tableau selon qu'ils sont inférieurs ou supérieur à une valeur x.

1. Choisir x parmi les éléments de t : donc il existe i ∈[0,n-1] tel que t[i]==x.
2. Séparer les éléments de t en 2 zones par rapport à x.
3. Placer x entre les 2 zones.

x est à la place qu'il occupera dans le tableau trié



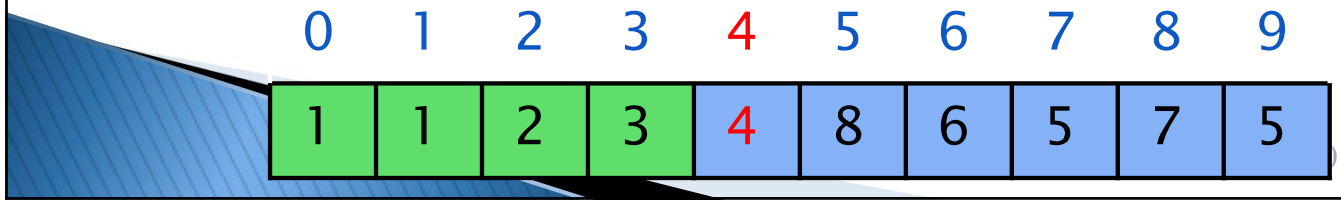
Fonction mystère

Idée :

- Utiliser la fonction `mystere` pour séparer les éléments d'un tableau selon qu'ils sont inférieurs ou supérieur à une valeur x.

1. Choisir x parmi les éléments de t : donc il existe i $\in$[0,n-1] tel que t[i]==x.
2. Séparer les éléments de t en 2 zones par rapport à x.
3. Placer x entre les 2 zones.

Combien de fois chaque élément de t est-il examiné ?



Fonction mystère

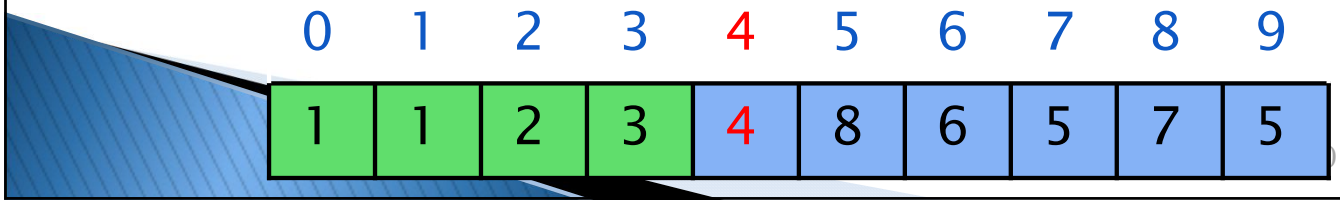
Idée :

- Utiliser la fonction `mystere` pour séparer les éléments d'un tableau selon qu'ils sont inférieurs ou supérieur à une valeur x.

1. Choisir x parmi les éléments de t : donc il existe i ∈[0,n-1] tel que t[i]==x.
2. Séparer les éléments de t en 2 zones par rapport à x.
3. Placer x entre les 2 zones.

Combien de fois chaque élément de t est-il examiné ?

une seule fois



Fonction mystère

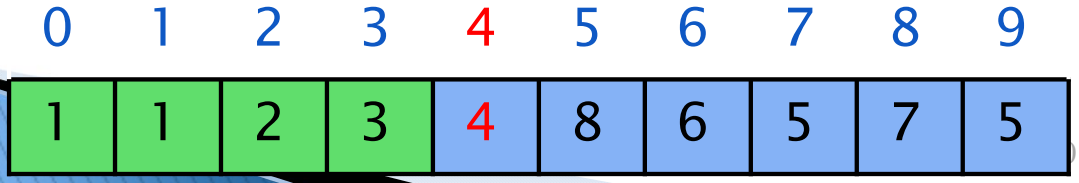
Idée :

- Utiliser la fonction `mystere` pour séparer les éléments d'un tableau selon qu'ils sont inférieurs ou supérieur à une valeur x.

1. Choisir x parmi les éléments de t : donc il existe i ∈[0,n-1] tel que t[i]==x.
2. Séparer les éléments de t en 2 zones par rapport à x.
3. Placer x entre les 2 zones.

x est à la place qu'il occupera dans le tableau trié

4. répéter 1, 2 et 3 sur chaque zone pour trier t récursivement



Fonction mystère

Idée :

- Utiliser la fonction `mystere` pour séparer les éléments d'un tableau selon qu'ils sont inférieurs ou supérieur à une valeur x.

1. Choisir x parmi les éléments de t : donc il existe i ∈[0,n-1] tel que t[i]==x.
2. Séparer les éléments de t en 2 zones par rapport à x.
3. Placer x entre les 2 zones.

x est à la place qu'il occupera dans le tableau trié

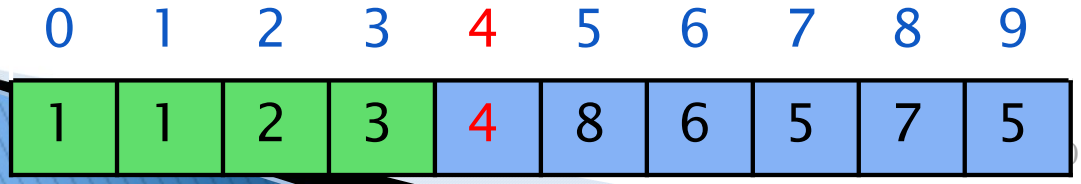
4. répéter 1, 2 et 3 sur chaque zone pour trier t récursivement

Question :

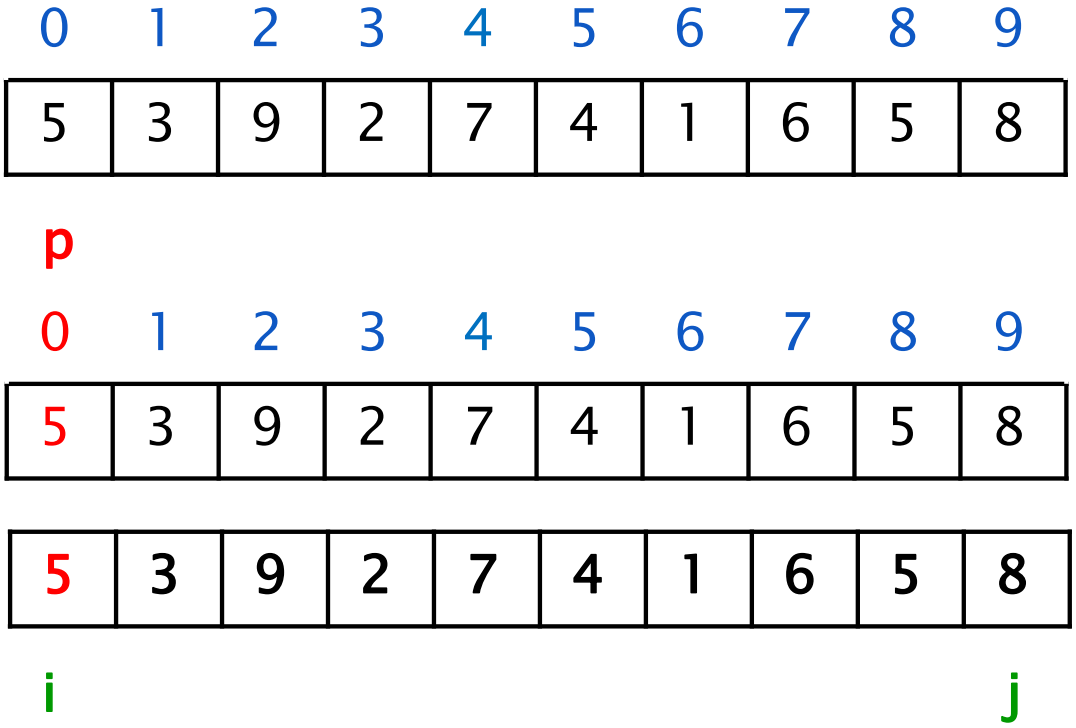
Comment choisir x ?

t[0] ou t[n-1] ou heuristique

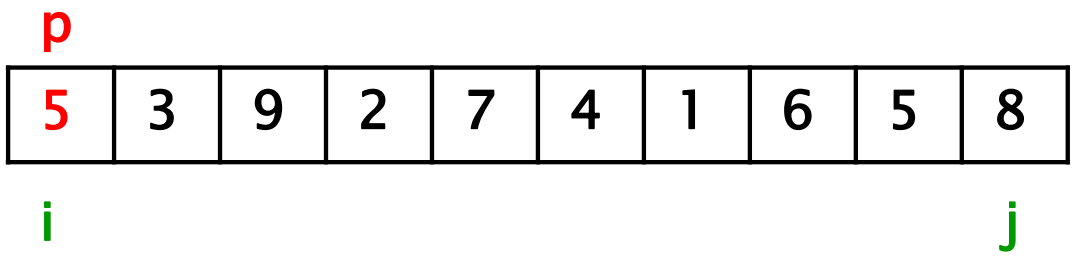
On appelle x le pivot



Exemple

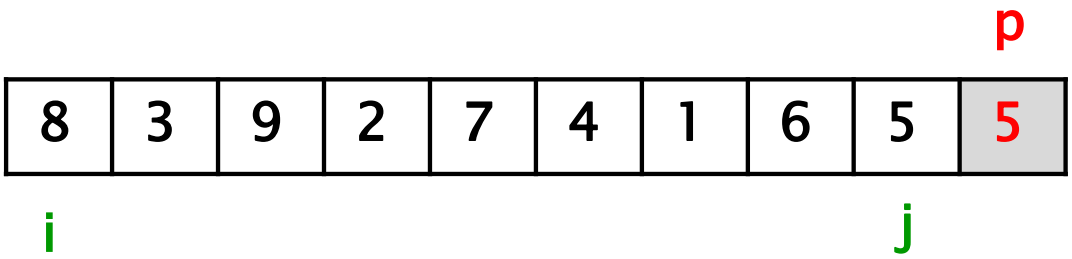

0 1 2 3 4 5 6 7 8 9
5 3 9 2 7 4 1 6 5 8
p
0 1 2 3 4 5 6 7 8 9
5 3 9 2 7 4 1 6 5 8
5 3 9 2 7 4 1 6 5 8
i
j

Exemple

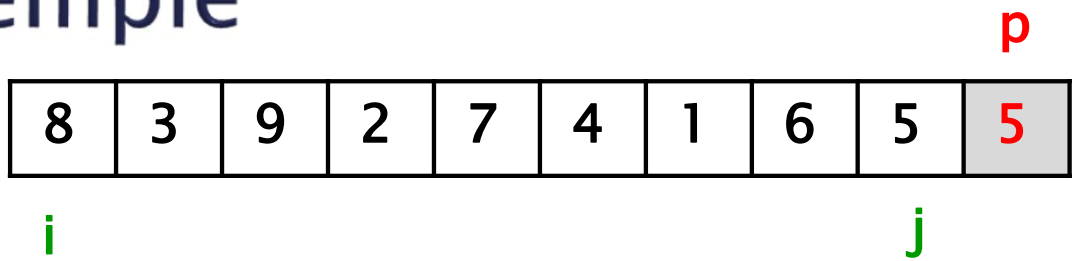



```
t[i]>=t[p] donc echanger(t,i,j)
```

Exemple

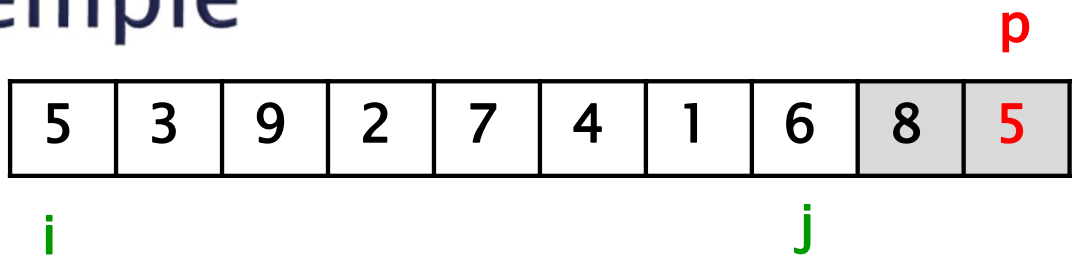


Exemple

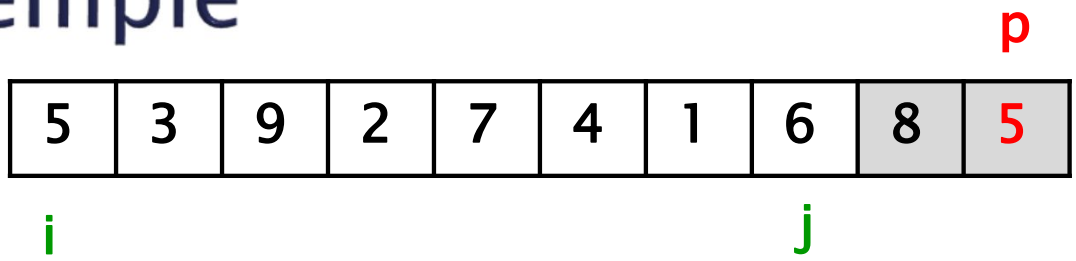



```
t[i]>=t[p] donc echanger(t,i,j)
```

Exemple

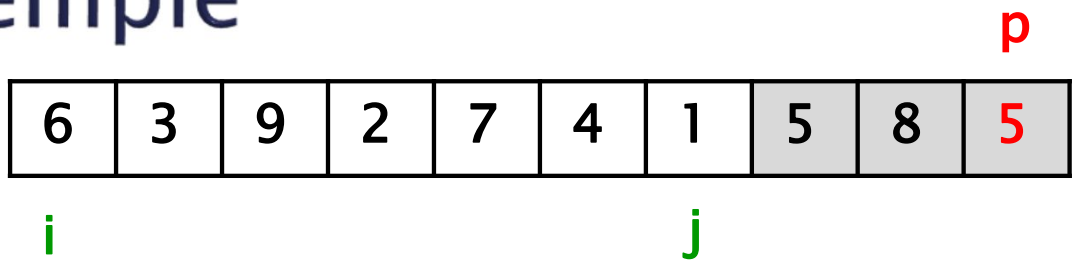


Exemple

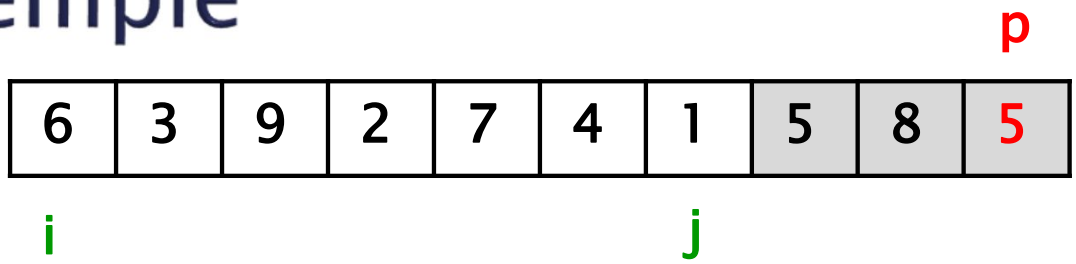



```
t[i]>=t[p] donc echanger(t,i,j)
```

Exemple

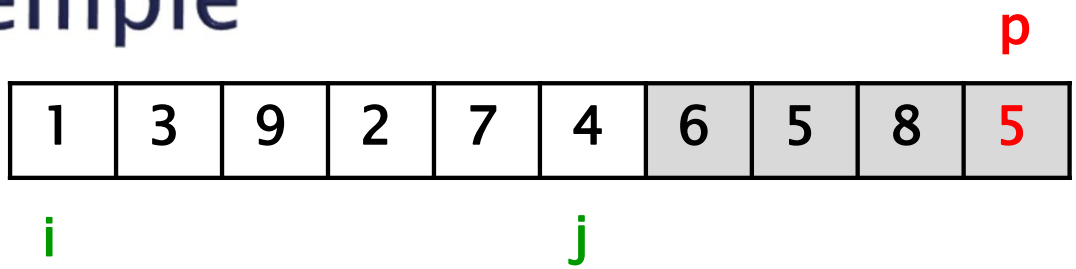


Exemple



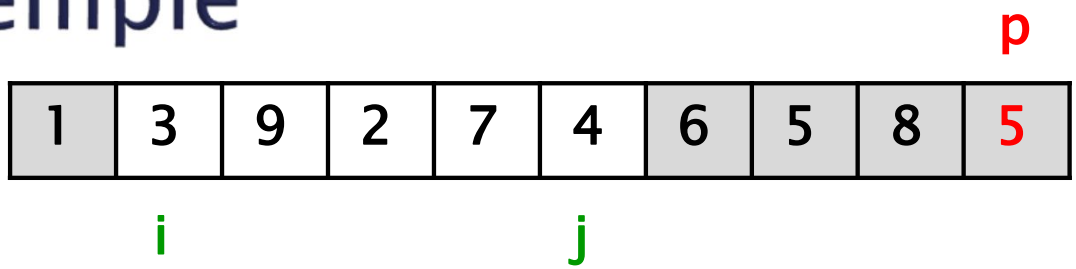

```
t[i]>=t[p] donc echanger(t,i,j)
```

Exemple

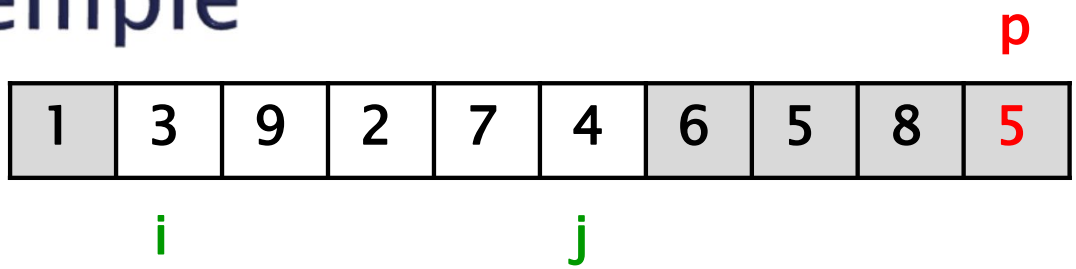


t[i]< t[p] donc pas d'échange

Exemple

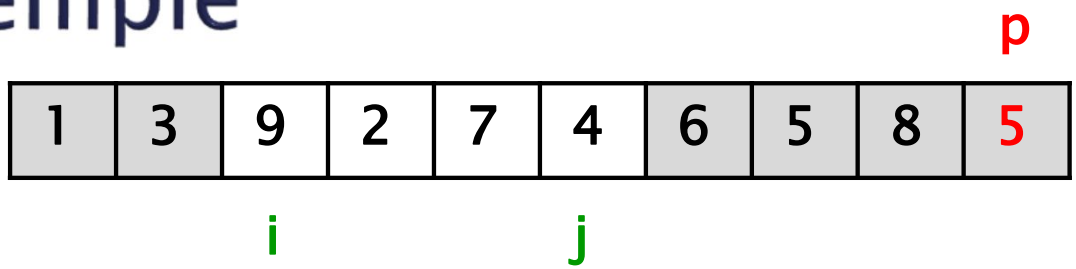


Exemple

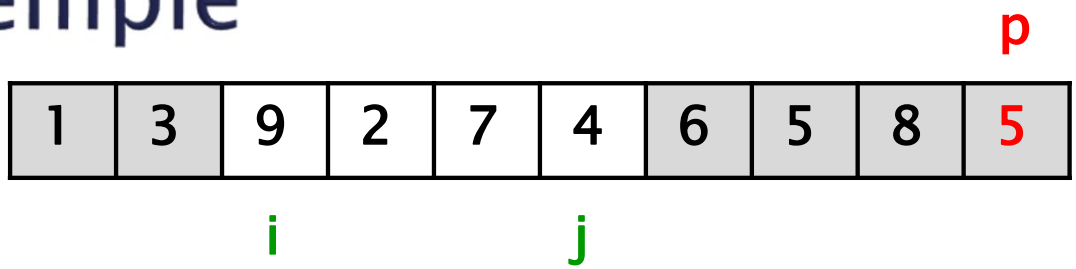


t[i]< t[p] donc pas d'échange

Exemple

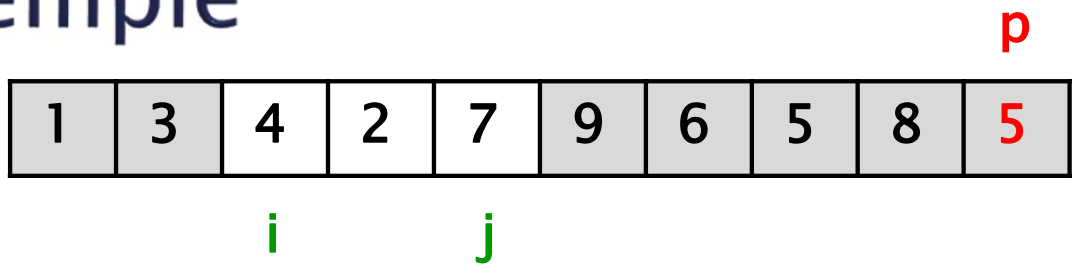


Exemple

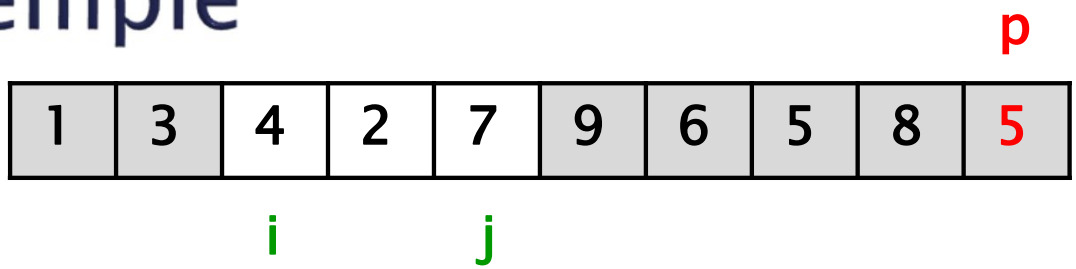



```
t[i]>=t[p] donc echanger(t,i,j)
```

Exemple

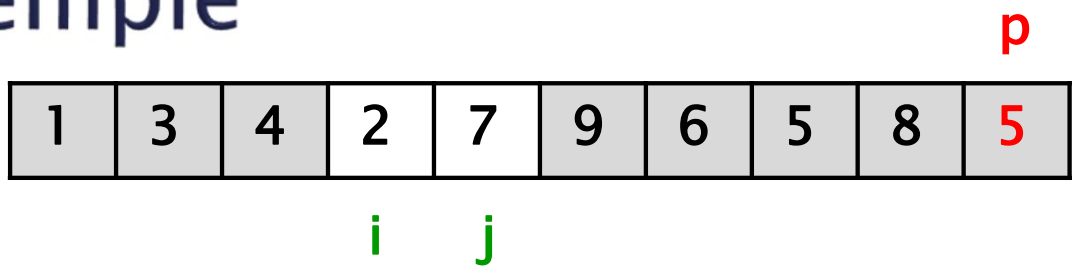


Exemple

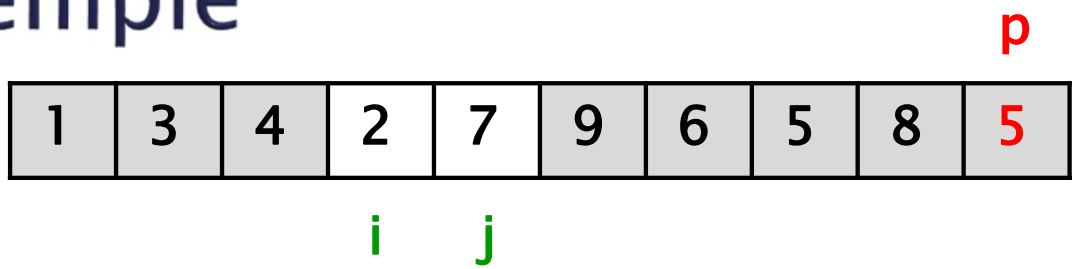


t[i]< t[p] donc pas d'échange

Exemple

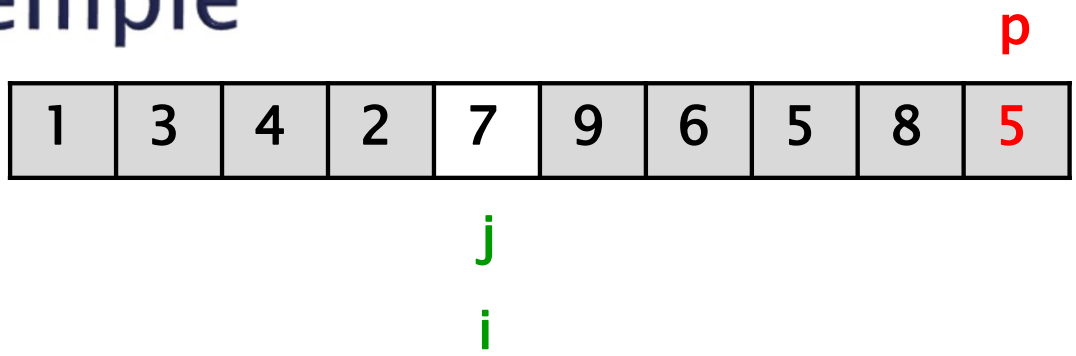


Exemple

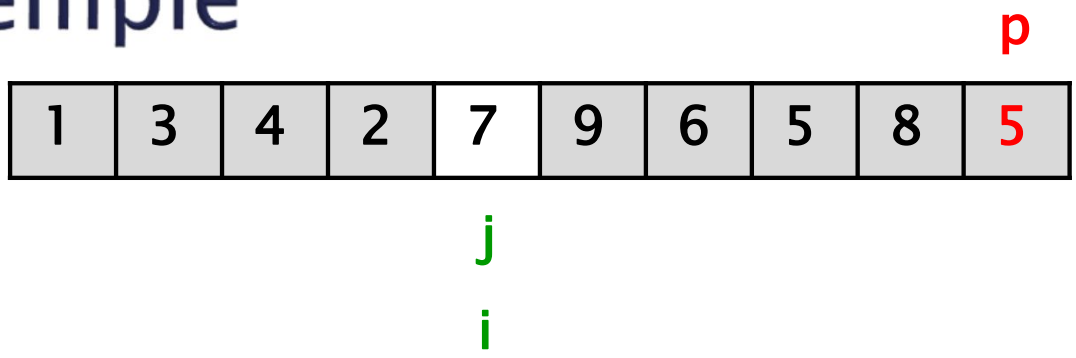


t[i]< t[p] donc pas d'échange

Exemple

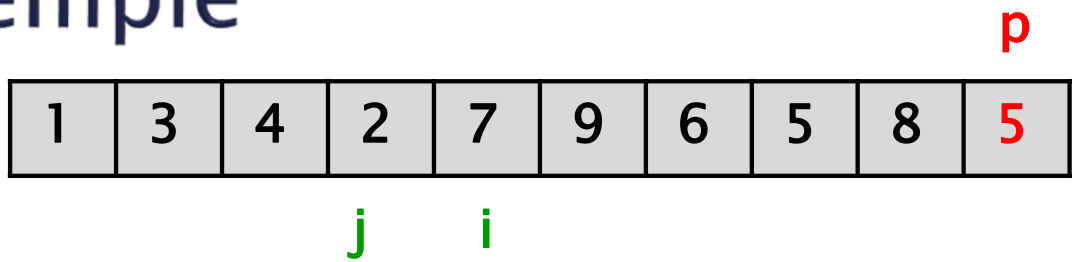


Exemple

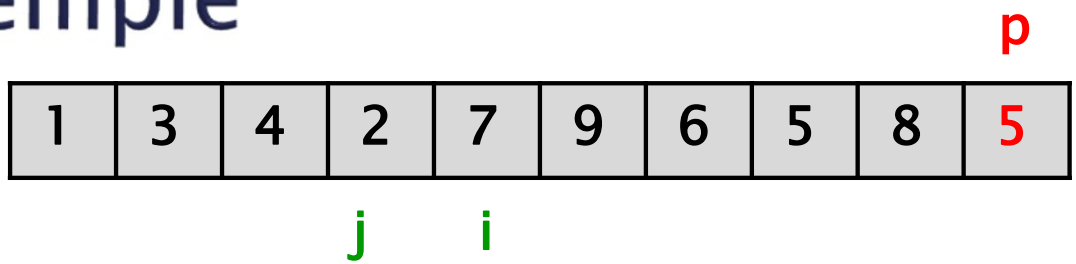



```
t[i]>=t[p] donc echanger(t,i,j)
```

Exemple

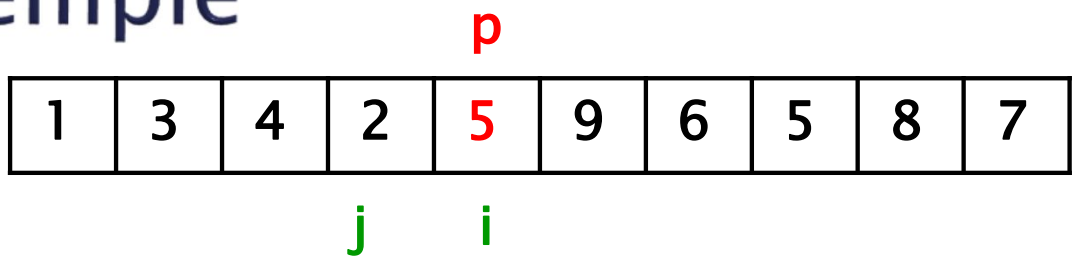


Exemple

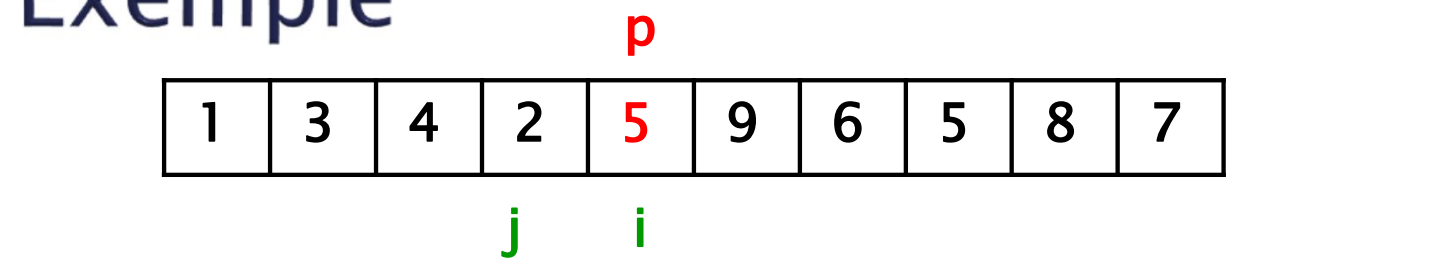


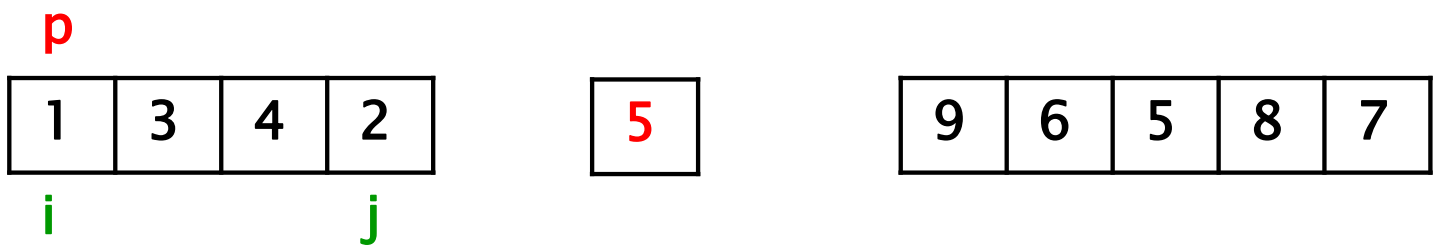

```
i>j donc placer t[p] à la place de t[i]
```

Exemple



Exemple





...

Séparation autour du pivot

```
def separationPivot(t,n, d, f):
  p = d
  i = d
  j = f
  while(i<=j):
    if (t[i]<t[p]):
      i=i+1
    else :
      echanger(t,i,j)
      if (i==p):
        p=j
      else :
        if (j==p):
          p=i
      j=j-1
  echanger (t, i, p)
  return (i)
```

En sortie le tableau t est réorganisé entre les indices d et f :

- d<=k<p, t[k]<t[p]
- p<=k<=f, t[k]>=t[p]
- t[p] est à sa place.
- Chaque élément a été examiné une seule fois
- Complexité en temps :

  O(f-d)
- Complexité espace :

  O(1)

Séparation autour du pivot

Il existe de nombreuses variantes de cette fonction :

- Heuristique pour le choix du pivot.
- Sans placement du pivot.
- Etc

Tri rapide

```
def triRapideRec(t, n, d, f):
  if(d<f):
    p=separationPivot(t,n,d,f)
    triRapideRec(t,n,d,p-1)
    triRapideRec(t,n,p+1,f)


def triRapide(t, n):
  triRapideRec(t,n,0,n-1)
```

- Complexité en temps : $\Omega(n\log_2(n))$ et $O(n^2)$
- Complexité en espace : $O(1)$
- Stabilité : non

Tris récursifs

Tri	Complexité en temps	Complexité espace	Stabilité
Fusion	$\Theta(n\log_2(n))$	$\Theta(\log_2(n))$ sur la pile $\Theta(n)$ en mémoire	Oui
Tri rapide	$\Omega(n\log_2(n))$ et $O(n^2)$	$\Theta(1)$	Non*

Tris rapide

La méthode pour choisir le pivot influence directement l'efficacité du tri rapide.

En moyenne le tri rapide est l'algorithme de tri le plus efficace.

Quicksort